VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe ve firmě
Individual Professional Practise in the Company

2011 Rudolf Kurka

Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Děkuji konzultantovi mé praxe Josefu Stromskému za velkou trpělivost při mém zaučování do práce v týmu a řediteli firmy Romanu Grycovi za osobní přístup.

Ve Frýdku-Místku dne 1. 4. 2011 ……

Rudolf Kurka

Abstrakt: Tato bakalářská práce se zabývá mojí odbornou praxí u firmy Elvac Solutions (člen skupiny Elvac, dřívější Elcom, sídlem Hasičská 53, Ostrava – Hrabůvka).
Začínám s krátkým seznámením s projektem infoPanels na kterém jsem odvedl nejvíc práce a uvádím příklady jeho reálného použití.
Následuje bližší popis firmy Elvac a jejích aktivit a úkolů které jsem zpracovával po dobu praxe včetně bližšího popisu aplikací v rámci infoPanels.
Poté se věnuji detailnějšímu popisu různých technologií, se kterými jsem přišel do styku a které jsem musel nastudovat, abych mohl splnit požadavky firmy na vývoj projektu.
Nakonec uvádím shrnutí a průběh praxe se zaměřením na nové poznatky a zkušenosti, které jsem nasbíral.

Abstract: This bachelor thesis covers my technical practice at the Elvac Solutions Ltd (member of the Elvac group, former Elcom, based Hasičská 53, Ostrava – Hrabůvka).
I'm beginning with brief description of the infoPanels project (which was my main concern during the thesis) and its usage and deployment examples.
Follows a closer look at the Elvac company and its activities and my tasks or objectives that I've been given in terms of development.
Then I go into detailed description of various technologies which I came across to and studied to accomplish development goals.
Thesis ends with summarization and practice progress with aim at new knowledge and experience I acquired.

Klíčová slova: .NET, Visual Studio, C#, MS SQL, Windows Presentation Foundation, Windows Communication Foundation, Entity Framework, Objektově relační mapování, Digital Signage

Keywords: .NET, Visual Studio, C#, MS SQL, Windows Presentation Foundation, Windows Communication Foundation, Entity Framework, Object Relational Mapping, Digital Signage

Seznam použitých zkratek a symbolů:

.NET Framework	- označuje sw vybavení platformy .NET na počítačích
ADO.NET	- ActiveX Data Objects .NET
API	- Application Programming Interface
CCTV	- Closed Circuit Television (uzavřený televizní okruh), kamerový sledovací systém
CLR	- Common Language Runtime
COM	- Component Object Model
DES	- Data Encryption Standard
Digital Signage	- systém digitálních zobrazovacích zařízení ve veřejných prostorách
DLL	- Dynamic-link library
EF	- Entity Framework
HDD	- hard disk drive
I/O	- vstup/výstup
IL	- Intermediate Language (dříve MSIL)
LINQ	- Language Integrated Query
RSS	- Really Simple Syndication
ORM	- objektově relační mapování
OS	- operační systém
UAC	- User Account Control
WCF	- Windows Communication Foundation
WMP	- Windows Media Player
WPF	- Windows Presentation Foundation
XAML	- Extensible Aplication Markup Language
XML	-Extensible Markup Language
XPO	- eXpress Persistent Objects
p/invoke	- Platform Invoke

Obsah

1 Úvod

1.1 Projekt infoPanels

InfoPanels jsou elektronické informační a reklamní panely, používané k prezentaci dynamických informací nezávisle na zobrazovacím zařízení - digital signage [1].

Systém infoPanels je složen z počítače s operačním systémem Windows, z příslušného softwarového vybavení a ze zobrazovací jednotky.

1.2 Komu je systém infoPanels určen

- Hotelům, bankám, kancelářským komplexům či prezentačním agenturám a školicím střediskům, které disponují konferenčními sály, salonky nebo zasedacími místnostmi a své prostory dynamicky využívají a pronajímají (pro snadnější orientaci klientů, hostů, či zákazníků při pořádání prezentací, přednášek či výstav).
- Muzeím, zdravotnickým zařízením, obchodním centrům, sportovním areálům.
- Dopravním společnostem (letiště, autobusová a vlaková nádraží) pro zobrazení provozních dat v dopravních uzlech.
- Průmyslovým společnostem pro zobrazení provozních dat, či orientaci v objektech (informační tabule ve výrobních halách, dispečinky, technické sály, CCTV).

1.3 Některá místa nasazení systému

- Intercontinental Praha
- Hotel Praha
- Holiday Inn Brno
- Harmony Club Hotel Špindlerův Mlýn
- Park Holiday Praha
- Kaštieľ Mojmírovce
- Letiště Ostrava
- INERGY Automotive Systems Slovakia
- ABB
- PEKAŘI a spol.
- Inergy Bratislava
- Hotel Savannah
- Třinecké železárny a.s.
- Club Hotel Praha
- DISTEP a.s.
- POSMEDIA s.r.o. (Electroworld)
- POSMEDIA Hungary (Electroworld)
- ADORE - propagační akce Ještěrka CUP - systém infoPanels zapůjčen pro vyhodnocování a zobrazování výsledků a prezentaci společnosti Linde Material Handling

2 Informace o firmě a činnosti studenta

2.1 Firma Elvac Solutions

Základním předmětem činnosti společnosti Elvac Solutions s.r.o. je poskytnutí řešení na klíč dle individuálních požadavků a potřeb zákazníka [2].

Společnost nabízí veškeré služby počínaje projektem, přes vývoj systému, jeho implementaci a konče servisní činnosti, a to zejména v oblastech:

- Monitorovací a řídicí systémy.
- Technicko-informační systémy (TIS).
- Informační displeje a navigační panely (InfoPanels).
- Geografické informační systémy (GIS).
- Monitorování datových sítí (WhatsUP).
- Webová řešení.
- Grafické práce.
- CitectSCADA.
- Vývoj software na zakázku.
- Systémová integrace.

2.2 Aktivity firmy

- Systém Flight Management použit při dnech NATO v Ostravě.
- Partner 12. Ročníku mezinárodní studentské soutěže GISáček v roce 2009.
- Ještěrka Cup 2009, k počítání a vyhodnocování výsledků byl použit systém infoPanels.
- A další….

2.3 Činnost a úkoly studenta

Hlavní část celé praxe se týkala projektu infoPanels. Jde o větší klient-server balík aplikací, který byl započat ještě na stařičkém .NET Frameworku 1.1 v roce 2003. Skládá se z cca 18 separátních řešení (solution), které reprezentují aplikace v rámci celého balíku. Řešení pak napočítají na 30 projektů psaných převážně v jazyce C#. Systém používá jeden hlavní server, na kterém jsou uloženy tzv. Layouty (prezentace určené ke spuštění na monitoru/televizi atd.) a media server na kterém jsou uloženy např. video soubory a zvukové soubory na které se layouty odkazují. Samotné layouty mají formu nastavitelné mřížky, na které uživatel umístí „playery“, tj. komponenty určené pro specifické účely, např. zobrazení navigačních šipek nebo obrázků, RSS zdrojů, informací o počasí nebo video/audio souborů atd.

S postupem let se na projektu vystřídalo mnoho generací programátorů s vlastními vizemi a idejemi, což se nevyhnutelně začalo projevovat na kvalitě kódu. Výsledkem bylo, že jsem začínal na projektu pro Visual Studio 2003, který byl na dnešní dobu velmi zastaralý a nešikovně napsaný.

Mým úkolem pak byl celý proces od restaurování projektu do novější verze Visual Studia, po upgrade stávající technologie, implementaci nových požadavků, až po průběžné ladění na základě odezvy zákazníků. Na celém projektu jsem od začátku dělal prakticky sám (co se týče samotného programování), ze standardního softwarového cyklu jsem byl ušetřen pouze nasazení nové verze. Při práci jsem si tedy vyzkoušel vývoj i podporu aplikace.

2.4 Některé hlavní aplikace v infoPanels

- Layouts Editor: Slouží k vytvoření a editaci souborů s infopanely, podporuje styl drag-and-drop na mřížku nastavitelné velikosti a upload souborů na media server.
- Pan Player: Slouží ke spouštění a prezentaci vlastních layoutů na vybraném zařízení.
- Pan Explorer: Slouží k organizaci dat na serveru, uploadu nových layoutů, svázání layoutů s konkrétním zobrazovacím zařízením, rozsáhlé konfiguraci např. časování spouštění layoutů, zobrazování splash screenů a další.

3 Vypracované úkoly a použité technologie

3.1 Upgrade zdrojových kódů

Prvním úkolem bylo se seznámit s projektem a začít s restaurací a oživením systému. Začal jsem s převodem balíku do Visual Studia 2008 a na platformu .NET 3.5. Většina projektů se po tomto zásahu nezkompilovala, takže začala ruční editace a refaktorizace zastaralé báze. Rozsáhlé úpravy se dotkly asi 35 tříd.

Poté jsem mohl začít se zběžným upgradem zdrojových kódů a algoritmů. Procházel jsem třídy a nastavení projektů a opravoval běžné a kritické chyby, které způsobovaly pád aplikace hned po spuštění. V úpravách jsem byl důsledný, aby bylo možné dále stavět na solidním základu. Dbal jsem na vyšší stupeň integrace s Visual Studiem jako je např. migrace dynamicky vytvářených komponent do designérů ve Visual Studiu. Úpravy se dotkly v podstatě všech hlavních a velké části vedlejších tříd v balíku. Opravoval jsem také soubory s prostředky (resources) plné neplatných odkazů a neexistujících položek a dále pak XML soubory s konfigurací kde byly např. nepoužívané connection stringy. Tyto úpravy jsem zpětně testoval, jednak z důvodu kontroly jestli nedošlo ke znehodnocení funkcionality (mnoho komponent v IP používá pozdní vazby) a jednak z důvodu využití optimalizací (sloučení connection stringů, prostředky které byly kdysi rozdílné, ale později došlo k jejich nahrazení za identické kopie jako zvuky, ikony atd.).

Jako další fázi jsem si naplánoval rozsáhlé refaktorizace s cílem výrazně zlepšit znovupoužitelnost, čitelnost a čistotu kódu a obecně modifikovat strukturu jednotlivých řešení tak, aby lépe vyhovovaly dnešním programovacím zvyklostem.

Pro představu o aktuální velikosti balíku uvádím hodnoty výstupu nástroje code-metrics Visual Studia pro hlavní projekty:

Projekt	Index udržovatelnosti	Cyklomatická složitost	Vnější reference	Počet řádků
DbConnect	78	116	59	496
InfoPanelsAboutBox	59	18	32	117
InfoPanelsCommon	82	4577	571	10946
LayoutsEditor	65	453	236	2356
LogAndMediaDelete	76	36	47	157
MediaSvc	66	72	39	209
PanExplorer3	78	2430	490	9656
PanPlayer2	88	787	206	2080
PanServer3Installer	76	59	26	176
PlayerLauncher3	82	408	158	1152
PlayerLauncherUpdater	80	47	58	180
StandaloneIP	74	243	124	644

Dále jsem doladil a vyčistil pre-build a post-build skripty, které téměř všechny obsahovaly chybu alespoň v podobě nesprávných souborových cest, poté jsem vytvořil hlavní řešení, do kterého jsem sjednotil všechny členské projekty, což značně usnadnilo a urychlilo kompilování jednotlivých buildů.

Opravil jsem desítky odkazů mezi projekty z referencí na DLL soubory na reference projektů, které Visual Studio podporuje, čímž se usnadní ladění projektů, zároveň jsem prošel celý strom projektových referencí a přidal nové tak, aby při změně zdrojového kódu byly automaticky kompilovány i všechny závislé projekty.

Do celého balíku jsem zavedl jednotkové testování (unit testing). Postupně jsem vytvořil testovací třídy pro vnitřní algoritmy a třídy, které jsem vyhodnotil jako kritické. Kromě těchto věcí jsem implementoval množství dalších minoritních úprav.

3.2 Kompatibilita s novými operačními systémy

První požadavek, který se týkal přidání nové funkcionality, byl zajistit bezproblémový chod infoPanels na systémech Windows Vista a Windows 7 a to i v 64bitové verzi. Začal jsem s testováním systému při běhu na 64bitové platformě. O nativním chodu nemělo smysl uvažovat, protože projekt je plný starších technologií v podobě COM komponent a DLL kompilovaných explicitně pro 32bitový systém. Pro tyto komponenty neexistují 64bitové verze, takže jsem všechny projekty uvnitř infoPanels nastavil na 32bitový režim. Bylo pak třeba aplikace zevrubně otestovat a odladit menší chyby s kompatibilitou.

Systém byl ovšem nadále nekompatibilní s bezpečnostním subsystémem UAC na Windows Vista a novějších. InfoPanels byl vystaven na klasickém Windows XP modelu kdy se např. nastavení a logy ukládají do souborů vedle hlavního .exe souboru aplikace. Model vyhovující UAC toto chování zakazuje (ProgramFiles je chráněný adresář a aplikace nemají implicitně právo zápisu do této složky oproti XP) stejně jako aplikuje množství dalších restrikcí hlavně na přístup do registru (virtualizace zápisu) atd.

Protože přizpůsobit infoPanels tak, aby plně vyhovoval tomuto modelu, by znamenalo přepsat prakticky veškerý I/O kód, rozhodl jsem se problém dočasně obejít vytvořením a integrací manifestů do všech spustitelných sestavení tak, aby byly spouštěny se zvýšenými (admin) právy, což deaktivuje řadu pravidel, které klade UAC. Tyto manifest soubory jsou rozpoznány operačním systémem a zajistí spuštění aplikace s admin právy nebo spuštění zabrání.

3.3 Logování

3.3.1 Log4net

Log4net je .NET port log4j frameworku, řešení pro programátory, kteří potřebují pokročilé logovací služby [3]. Vystavuje jednoduché i komplexní metody pro vkládání záznamů do logu a nabízí rozsáhlé možnosti konfigurace, např. ukládání logu na server nebo mail.

3.3.2 Řešení úkolů

Systém infoPanels v podstatě nepoužíval logování, což bylo zcela nedostačující zvláště v případech pádu aplikace. Většina chyb se propagovala jen ve formě textových zpráv zobrazených uživateli, to byla velká překážka při ladění a to hlavně u nasazených verzí. Bylo tedy třeba implementovat logovací systém, který bude schopný pojmout a zpracovat alespoň obvyklé záznamy typu upozornění, výstraha a chyba a zajistí jejich distribuci na patřičná místa.

Dostal jsem za úkol požít komponentu log4net protože byla v jednom projektu již z části integrovaná. Nastudoval jsem manuály ke komponentě a postupně jí nahradil nebo vylepšil stávající systém u mnoha kritických tříd.

Log4net jsem pak integroval do balíku infoPanels i nadále během praxe jak jsem postupoval s vývojem a nacházel další místa, kde bylo vhodné využít logování.

Toto logování se později velmi osvědčilo hlavně při vzdálené podpoře zákazníků. Mnohdy jsem např. vytvořil konfigurační soubory tak, aby log4net zasílal kritické chyby na vyhrazený firemní server apod.

3.4 Komunikace mezi procesy

3.4.1 Windows Communication Foundation

Windows Communication Foundation (WCF) představuje rámec pro vytváření aplikací orientovaných na služby [4]. Pomocí WCF je možné posílat data jako asynchronní zprávy z jednoho koncového bodu do druhého. Koncový bod služby může být součástí nepřetržitě dostupné služby hostované skrze IIS, nebo může zastupovat službu hostovanou v aplikaci. Koncový bod může být klient služby, která přijímá data koncového bodu jiné služby. Zprávy mohou být jednoduché jako jediný znak nebo slovo, které jsou odeslány jako XML, nebo složité, jako jsou datové proudy binárních dat. Některé typické scénáře:

- Zabezpečené služby ke zpracování obchodních transakcí.
- Služba, která poskytuje aktuální data ostatním uživatelům, jako je například dopravní informace nebo jiné monitorovací služby.
- Chatovací služba, která umožňuje dvěma lidem komunikovat nebo vyměňovat data v reálném čase.
- Aplikace obsahující informační nástěnky/tabule, která dotazuje více služeb pro data a prezentuje je v logickém sledu.
- Vystavuje workflow implementovaný pomocí Windows Workflow Foundation jako WCF služba.
- Silverlight aplikace dotazující služby pro nejnovější data.

3.4.2 Řešení úkolů

Vedení projektu mi dalo za úkol přepracovat způsob, jakým mezi sebou komunikují aplikace z balíku. To se týká asi 5 projektů. Nejpatrnější je komunikace mezi Pan Playerem, který zobrazuje uživatelské layouty a Player Launcherem, který slouží jako „watchdog“ a zodpovídá např. za načasované spuštění, restart v případě chyby, ukončení nebo i update Pan Playeru.

Předcházející způsob „komunikace“ mezi jejich procesy fungovala tak, že Player Launcher vytvořil lokální soubor s napevno určeným jménem a Pan Player si tento soubor po spuštění má přečíst a vytáhnout z něj konfiguraci. Takový způsob je velmi nedostačující a má řadu nevýhod:

- Celkově pomalé.
- Zbytečné vyžádání HDD jako komunikačního bufferu.
- Předpokládá se přístup pro zápis do lokálního souboru což je jasná překážka pro UAC protože program je spuštěn v chráněném/vizualizovaném adresáři (ProgramFiles).
- Potenciálně nebezpečné, v souboru se ukládaly connection stringy i s hesly, soubor se sice šifroval, ale s dnes již prolomenými algoritmy (jednoduchý DES), navíc se soubor celkem zbytečně komprimoval, což prodloužilo čas pro výměnu informací až do řádu sekund.
- Velké množství asociovaného kódu na obou stranách.

V ostatních projektech byla situace obdobná, rozhodl jsem se ji řešit pomocí pojmenovaných kanálů (named pipes) jakožto plnohodnotného inter-process komunikačního způsobu. Je také součástí Windows, takže není třeba integrovat do projektu nové komponenty. Na platformě .NET 3.5 a vyšší se práce ještě dále ulehčuje díky technologii Windows Communication Foundation. Tu jsem nakonfiguroval ke klient-server komunikaci mezi instancemi aplikací z infoPanels přes pojmenované kanály na lokálním počítači, což celou magii v pozadí pohodlně zapouzdřilo do jednoho aplikačního rámce.

Stávající systém komunikace mezi projekty jsem pak kompletně předělal. Vytvořil jsem kontrakty služeb pro předávání konfiguračních objektů a přizpůsobil této formě komunikace aplikace v balíku.

Teď mezi sebou procesy komunikují mnohem svižněji a efektivněji. Také došlo ke drastickému zredukování asociovaného kódu z několika stovek na několik desítek řádků. Vedle toho je teď možná plně obousměrná komunikace čehož jsem později využil při implementaci nových požadavků.

3.5 Vývoj nových komponent

V pozdější fázi praxe jsem dostal za úkol naprogramovat několik komponent v podobě samostatných playerů pokud možno tak, aby byly samostatně licencovatelné.

3.5.1 Přehrávač médií

3.5.1.1 Windows Presentation Foundation

Windows Presentation Foundation (WPF) je mocná knihovna tříd pro vytváření bohatých uživatelských rozhraní, kterou je netřeba příliš představovat [5]. Používá deklarativní styl pro definici rozhraní, které se ukládají do souborů .xaml. Lze však také použít čistě programátorský a dynamický přístup. Představuje revoluci v přístupu k vytváření uživatelských rozhraní.

3.5.1.2 Řešení úkolů

První komponentou byl přehrávač multimediálních souborů a PowerPoint prezentací. V projektu již byla komponenta pro přehrávání videa, ale byla téměř nefunkční. Pod pokličkou to byl jen Windows Media Player ovládaný přes ActiveX knihovnu importovanou v projektu. To bylo provedeno před dávnými časy pro archaickou verzi WMP a podle toho pak vypadalo i interop sestavení generované Visual Studiem.

Jelikož jsem na platformě .NET 3.5 měl k dispozici WPF, rozhodl jsem se celou komponentu zrušit a napsat znova. Pro video a audio vstup jsem použil WPF komponentu MediaElement. Jak jsem později zjistil později při studiu vnitřního fungování WPF, tato komponenta také využívá Windows Media Player, nicméně dělá to korektní cestou a je kompatibilní s novými verzemi. Také tradičně schovává zdlouhavou a nešikovnou logiku práce s externími COM a ActiveX komponentami do sady pohodlných funkcí.

Aby bylo možné umístit WPF prvek na WinForms formulář, bylo nutné nejprve zavést hostování WPF skrze prvek ElementHost atd. Přehrávač PowerPoint prezentací jsem převedl z jiného playeru, který jsem tak mohl zrušit.

Tento nový přehrávač reagoval nesrovnatelně rychleji a to tak, že jsem dokonce později mohl implementovat jeho živý náhled. Také opět došlo k obrovskému zredukování kódu.

3.5.2 Přehrávač flash animací

3.5.2.1 Flash in box

Flash in box je jedna ze zajímavých pomocných technologií, které jsem měl za úkol integrovat do infoPanels [6]. Jedná se o stand-alone přehrávač všech druhů flash souborů. Je stavěná jako UserControl komponenta a pohodlně se integruje mezi ostatní prvky Windows Forms 2.0.

Charakteristiky komponenty:

- Načítání flash souborů bez externích kodeků (dokonce i bez nainstalovaného Flash Playeru).
- Podporuje průhlednost.
- Přehrávání streamovaných flash souborů.
- Vystavuje některé funkce Flash API.

3.5.2.2 Řešení úkolů

Dlouho si zákazníci stěžovali na nemožnost přehrávat flash prezentace. V případech, kde se nepředpokládá možnost vytváření složitých videoprezentací, jako např. kinoreklamy a navigační nebo reklamní displeje se vskutku jedná o snadno použitelnou a spolehlivou technologii, kterou je snadné ovládnout.

Proto mi vedení jako další úkol dalo vytvořit player pro prezentaci flash animací. Měl jsem využít komponenty flash-in-box, protože již byla využita pro jiné účely na jiném místě v infoPanels. Firma zakoupila novou verzi, kterou jsem nahradil stávající roky starou a použil jsem ji k implementaci jednoduchého flash přehrávače. S grafikou kolem (ikony, náhledy) mi pomohl tým grafiků ve firmě.

3.5.3 Přehrávač PDF

3.5.3.1 PDFLibNet

PDFLibNet je součást open-source projektu pdfviewernet [7]. Vystavuje Windows Forms komponentu ve které je možné zobrazovat PDF soubory. Hlavní předností je, že komponenta nevyžaduje nainstalovaný Acrobat Reader a je zdarma.

3.5.3.2 Řešení úkolů

Další player na žádost zákazníka. Komponentu pro zobrazení PDF jsem zvolil PDFlibNet protože je jako jedna z mála zdarma i pro komerční účely. Po naprogramování testovací aplikace s ní vedení souhlasilo a dále jsem postupoval obdobně jako v případě flash playeru.

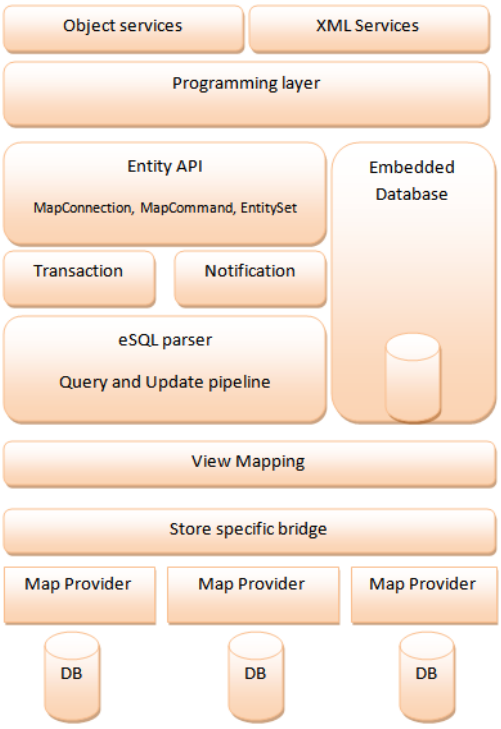
3.6 Reimplementace datové vrstvy

3.6.1 Entity Framework + LINQ to Entities

Jedná se o odpověď Microsoftu na databázové programování v .NET [8]. Zajišťuje objektově relační mapování na bázi ručně konfigurovatelných XML souborů.

Další výhody EF:

- Nezávislost na databázi. EF je podobně jako LINQ založena na providerech, tzn. je možné doinstalovat např. providera pro Oracle (3rd party). Nativně EF podporuje pouze MS SQL.
- Kontrola nad lazy-loading mechanizmem, což umožní optimalizace výkonu. Pomocí metody Load() záznamu explicitně načteme referencované záznamy z databáze (explicit-loading), metoda Include() u EF tabulky se dá také použít (eager-loading), ale vyžaduje textové názvy tabulek, což může být nevýhodné při propagování případných změn databáze do programu. EF verze 4.0 pak podporuje i implicitní lazy-loading nastavitelný přímo přes designér nebo programově v EFContext.ContextOptions.LazyLoadingEnabled.
- Plná podpora LINQ dotazů.
- Podporuje import i uložených procedur (opět typově bezpečné).
- Integrovaný designér ve Visual Studiu.
- Plná podpora intellisense v kódu.
- Extrémně rychlý vývoj v porovnání s konkurencí.



Obrázek 1: Struktura Entity Framework vrstvového modelu

3.6.2 Řešení úkolů

Systém infoPanels je hustě protkán databázovým kódem. Většina formulářů zobrazuje nějaká data ze serveru, ať už uživatelská nastavení, layouty (ukládají se celé na serveru), dostupné komponenty pro vytváření prezentací, licence nebo také soubory s updaty programu. Téměř veškerá hlavní funkcionalita je tak v nějakém bodě závislá na komunikaci s databázovým serverem.

Server je klasického typu s Microsoft SQL Server 2005/2008 databází, media server je potom webová služba. Jelikož si původně aplikace v infoPanels zařizovaly vlastní přístup k databázi (téměř nulová abstrakce, mnoho formulářů si zbytečně vytvářelo a udržovalo spojení samo), síťový kód postupně narostl do gigantických rozměrů o tisícovkách řádků. Předchozí generace programátorů si navíc chtěly šetřit práci a tak velká část kódu byla buď úplně totožná (pouze zkopírovaná do několika sestavení/projektů), nebo alespoň vykonávala stejnou činnost nad jinými daty. Výsledkem bylo, že aplikace byly velmi neresponzivní, často se na dlouhou dobu zasekly úplně nebo spadly při přerušení spojení se serverem (to se týkalo hlavně editačních aplikací).

Navíc kód byl značně zastaralý, databáze byla reprezentována jednoduchými datasety (ještě jen velmi povrchně nakonfigurovanými) a věci jako např. vygenerování nové hodnoty primárního klíče se dělaly ručně v kódu i přes to, že se jednalo o číselnou sekvenci. Databáze sama pak nebyla nijak dobře navrhnuta. Mnoho tabulek obsahovalo redundantní odkazy, nepoužívaly se uložené procedury, i když od pohledu tu pro ně bylo mnoho místa atd.

Když mi vedení dalo za úkol, abych zlepšil tuto situaci, rozhodl jsem se celý databázový kód přepsat a patřičně modernizovat. Při příležitosti modernizace jsem také chtěl přejít na nějaké novější vysokoúrovňové ORM a pokud možno se vyhnout ručnímu přístupu přes ADO.NET. Firma mi dala na výběr mezi Entity Framework od firmy Microsoft a XPO od Developer Express. Zvolil jsem technologii Entity Framework z několika důvodů:

- Plně integrována do aktuální verze .NET Frameworku.
- Je k dispozici zdarma.
- Pozice na trhu by měla být do budoucna zajištěná (oproti XPO).
- Propracovaná integrace s Visual Studiem.
- Hodí se do RAD stylu vývoje.
- Nebyly absolutně žádné plány na přechod na jakýkoliv jiný server než Microsoft SQL Server, tudíž nevadila často vytýkaná specializace Entity Frameworku (pro např. Oracle providera by bylo nutné zakoupit a integrovat software třetích stran).

Po výběru technologie jsem již mohl začít s revizí samotné databáze. Vygeneroval jsem si logický model databáze (databáze je spíš jednoduchá s 8 tabulkami) a izoloval přebytečné atributy, po této optimalizaci jsem se zbavil jedné vazební tabulky. Pak jsem začal s vlastním přepisováním, procházel jsem kód a mapoval strukturu databázových operací, přitom jsem souběžně vytvářel množství uložených procedur, které později ušetřily mnoho práce v kódu. Velké refaktorování nastalo po smazání datasetů. Vytvořil jsem model databáze a začal přepisovat databázový kód metodu po metodě. Všechen kód jsem také přesunul do vlastního sestavení jako korektní reprezentaci datové vrstvy pro celý balík.

Trvalo dlouho než šel balík opět zkompilovat, ale výsledek stál za to. Prakticky všechny aplikace se viditelně zrychlily, velmi výrazně se usnadnila implementace nové funkcionality a ušetřily se stovky řádků složitého kódu (většina kódu je stejně spravována grafickým designérem Entity Frameworku), aplikace jsou teď také snadno konfigurovatelné jak z programu tak zvenčí editací XML souboru s nastavením vrstvy zajišťující abstrakci databáze.

3.7 Ladění

3.7.1 p/invoke a.k.a platform invoke

Systém infoPanels do velmi velké míry spoléhá na Windows API. V prostředí .NET se tedy automaticky nabízí technologie p/invoke. Jedná se o v celku rozsáhlý nástroj umožňující volat funkce v komponentách napsaných v některých jazycích jiných než .NET, např. nejčastěji komponenty napsané pomocí technologie COM, na čemž stojí celé tzv. Win32 API.

Běžný způsob použití spočívá v deklarativní definici importu funkce (včetně mapování typů pro parametry a návratové hodnoty). Tato naimportovaná funkce se pak používá jako běžná statická funkce v .NET a potřebnou magii v pozadí (lokalizace knihovny, napojení na rozhraní COM, lokalizace hledané metody, převody hodnot mezi prostředími atd.) zajistí .NET běhové prostředí (CLR). Hlavní (notoricky známá) třída pro p/invoke se nazývá DllImport a používá se jako atribut.

```
[DllImport("user32.dll")]
[return: MarshalAs(UnmanagedType.Bool)]
internal static extern bool GetWindowRect(IntPtr hWnd, out RECT lpRect);
```

3.7.2 Řešení úkolů

Velká část praxe byla věnována náročnému bug huntingu. Mnoho zákazníků si stěžovalo na chyby, které předchozí generace programátorů nebyly schopné odstranit. Jednalo se veskrze o velmi obtížně reprodukovatelné pády aplikace. Další důležitý faktor byl, že infoPanels jsou mnohdy provozovány jako kritický systém, například pro LCD panely které v případě ohrožení zobrazují únikové informace a požární směrnice. Systém tedy musí splňovat určitý standard stability. Moje úkoly tak byly najít a eliminovat co nejvíc starých, těžko identifikovatelných chyb a odzkoušet dlouhodobou stabilitu.

Testování jsem prováděl na firemním stroji s plazmovým panelem, což jsou obvyklé reálné podmínky. Chyby měly charakter náhodných pádů aplikace a symptomy byly většinou podobné, na určitých HW konfiguracích docházelo k nepředvídanému pádu aplikace bez záznamu v logu. Spustil jsem na testovacím PC profilovací prostředí (ANTS profiler) a začaly hodiny víceméně bezcílné práce v programech (ve snaze pokrýt spouštěcí mechanizmy chyb), vyhodnocování profilovacích výsledků a nakonec vlastního ladění. Tyto úkoly patřily k nejobtížnějším za celou praxi a prověřily moji mnohaletou praxi v .NETu. Na několika místech jsem dokonce byl nucen pracovat na úrovni IL a procvičil si tak typické assemblerovské ladění.

Mravenčí prací jsem se po malých krůčcích dostával k jádru chyb. Typově se (většinou) jednalo o úniky neřízené paměti procesu, které způsobily pád aplikace v náhodných intervalech. Řádek po řádku jsem hledal a profiloval zodpovědné metody, které byly obvykle zanořeny hluboko v zásobníku volání.

Nakonec se mi podařilo identifikovat hned několik slabých míst, kde jsem rozhodl, jestli kód reimplementovat nebo funkcionalitu dočasně zrušit (hladký chod aplikací byla priorita). Podle očekávání viníky úniků paměti byly většinou p/invoke funkce. Podařilo se mi zjistit 3 takové. Jednu jsem opravil, problémem bylo známé neuvolnění paměti po práci s neřízenou bitmapou. Další se týkala pointerů na .NET okno formuláře což šlo snadno nahradit řízeným ekvivalentem. Poslední se byla v interop wrapperu pro COM knihovnu generovaného předcházející, starou verzí Visual Studia, sestavení stačilo nechat znovu vygenerovat novějším Visual Studiem. Po těchto úpravách již k pádům téměř nedocházelo a s laděním jsem pokračoval do konce praxe.

4 Závěr

Na konci praxe jsem splnil všechny požadavky a firma byla s výsledky spokojena. Největší přínos pro mě byla určitě práce v týmu. Zatímco implementaci jsem dělal spíš samostatně, tak třeba organizace a podpora tak velkého projektu pro mě byla neznámá.

Chybějící znalosti se většinou týkaly neznalosti některých okrajových technologií. Na platformě .NET programuji již desátým rokem, takže jsem neměl žádné velké mezery, které bych nebyl schopný rychle dohnat a práce ač místy mravenčí, mě velmi bavila.

Pro spokojenost na straně firmy nadále zůstávám zaměstnán a pokračuji práce na infoPanels přičemž se chystá započetí zbrusu nové verze na platformě .NET 4.0, pro kterou jsem hlavní programátor.

Literatura

[1] *Homepage infoPanels.* [Online] http://www.elvacsolutions.eu/index.php/cs/produkty-a-sluzby/infopanels.

[2] *Elvac Solutions s.r.o.* [Online] http://www.elvacsolutions.eu.

[3] *Homepage log4net.* [Online] http://logging.apache.org/log4net/index.html.

[4] *MSDN pro WCF.* [Online] http://msdn.microsoft.com/en-us/library/ms735119(VS.90).aspx.

[5] *MSDN pro WPF.* [Online] http://msdn.microsoft.com/en-us/library/ms754130.aspx.

[6] *Homepage Flash in Box.* [Online] http://www.f-in-box.com/dotnet.

[7] *Homepage PDFLibNet (PDFViewer.NET).* [Online] http://code.google.com/p/pdfviewernet/.

[8] *MSDN pro EF.* [Online] http://msdn.microsoft.com/en-us/library/aa697427(VS.80).aspx.

[9] Nagel Ch., Evjen B.,Glynn J.,Skinner M.,W. *C# 2008 - Programujeme profesionálně, (2. vydání).* Computer Press, 2009, ISBN 9788025124017.

[10] Charles, Petzold. *Mistrovství ve Windows Presentation Foundation.* Computer Press, 2008, ISBN 9788025121412.

[11] *MSDN nápověda.* [Online] http://msdn.microsoft.com/en-us/library/default.aspx.